EXHIBIT 21

CISCO SYSTEMS

Document Number	ENG-29568
Revision	D
Author	Dan Tappan
Project Manager	Pat Monk

IOS CLI for BGP/MPLS VPN

Headline

This document describes the IOS CLI commands and the configuration model for supporting BGP/MPLS VPNs

Rev.	Date	Originator	Comment
A,B	11/98	Dan Tappan	Initial version
C	2/98	Dan Tappan	Update for multiple communities
D	3/98	Dan Tappan	Update for 'ip vrf' mode. Add note to Hub/spoke example

Modification History

Definitions

VPN Provider	A Service provider who sells a Virtual Private Network service
PE	Provider Edge Router. A Provider router which connects to VPN customer sites
CE	Customer Edge Router. A customer router in a VPN site, which connects to a PE router
VPN	Virtual Private Network. A set of customer sites which are allowed to communicate through a VPN service
VRF	VPN Routing/Forwarding table. A Routing table instance which is populated with VPN routes.
VPNv4 NLRI	A BGP NLRI which corresponds to a VPN IPv4 prefix.
RD	Route Distinguisher. An 8 byte value value which is prepended to an IPv4 prefix to create a unique VPN IPv4 prefix

1.0 Introduction

This document describes the IOS configuration commands, in the first EFT image of the VPN code, which support the Cisco BGP/MPLS VPN service.

2.0 Concepts

Cisco BGP/MPLS VPN allows a service provider to implement a Virtual Private network service using a Layer 3 backbone.

2.1 Model

The BGP/MPLS VPN architecture is modeled in Figure 1, .

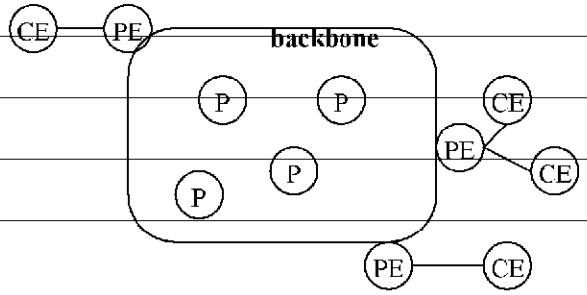

Figure 1, BGP/MPLS VPN model

This is made up of the following components:

- A Service Provider backbone network.
- The Provider backbone is made up of Provider (P) routers. These routers are not VPN aware, but provide a transport service between the edge routers.
- At the edge of the provider network are Provider Edge (PE) routers. These routers maintain customer VPN routing databases. All of the VPN specific mechanisms are implemented by the PE routers.
- At the edge of the customer network is a Customer Edge (CE) router. This router is not VPN aware.

BGP/MPLS VPN is implemented through the following mechanisms:

- Multiprotocol BGP extensions (RFC2283) are used to encode customer address prefixes into unique control plane NLRIs, as defined in Section 2.5.
- Extended BGP community attributes are used to flexibly control the distribution of customer prefixes.
- Associated with each customer route is an MPLS label. This is assigned by the PE router which originated the route, and is used to direct data packets to the correct CE router.
- MPLS forwarding is used across the provider backbone, based on either dynamic IP paths, or Traffic Engineered paths.
- Thus, when a customer data packet is forwarded across the backbone, two levels of label are used. The top label directs the packet to the correct PE router. The second label indicates how that PE should forward the packet.
- Cisco Tag Cos mechanisms provide service differentiation between customer data packets.
- Standard IP forwarding is used between the PE and CE routers. The PE associates each CE with a forwarding table which contains only the set of routes which should be available to that CE. This mechanism is discussed further in Section 2.3.

2.2 VPNs

Architecturally, a Virtual Private Network is defined as a set of sites which share a common community of interest. That is, a set of sites which are allowed to communicate across the VPN backbone. For example, Figure 2, defines 5 sites, and 3 VPNs.

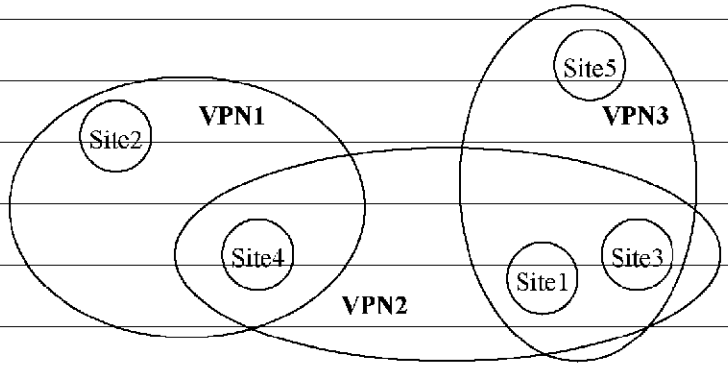

Figure 2, VPN model

- VPN1 consists of Site2 and Site4
- VPN2 consists of Site4, Site1, and Site3
- VPN3 consists of Site5, Site1, and Site3

Virtual Private Networks have the following properties:

- Sites within a VPN can use private (RFC1918) addressing. There is no requirement that an address be global across the entire set of sites.
- Sites which are a member of a VPN can communicate through the service provider backbone. Sites which are not members of a common VPN cannot communicate.
- Isolation is provided through constrained delivery of routing information, and through distinct forwarding tables.

2.3 VPN Routing/Forwarding instances

To support the architectural concept of a VPN, IOS implements multiple distinct VPN Routing/Forwarding instances: sometimes referred to as VRFs.

A VRF consists of:

- An IP Routing table. This is directly analogous to the single central routing table in previous versions of IOS.[1]
- A derived forwarding table, based on the CEF forwarding technology
- A set of interfaces which use the derived forwarding table
- Rules which control which routes are injected into a VRF routing table, and which routes may be exported to other VRF instances.
- A set of routing protocols and routing peers which inject information into the VRF routing table
- router variables associated with the VRF routing instance.

Thus, for the example in Figure 2, we have four distinct sets of routing information:

- Site2 which only gets routing information for VPN1
- Site4, which get routing information for both VPN1 and VPN2
- Site3 and Site1, both of which get routing information for both VPN2 and VPN3

1. In fact, the central routing table can be considered to be the VRF routing table for the provider backbone.

- Site5 which only get routing information for VPN3

Assuming all of these sites are attached to a single PE router, each of these sets corresponds to a VRF.

2.4 Router Address Families

The IOS BGP/MPLS code supports a general mechanism for configuring routing protocols which can carry multiple Level 3 protocols. To support this, we introduce the concept of a router "address-family". In general, an address family consists of a main family (also referred to as an Address Family Identifier, or AFI) and a modifier, or SubAddress-Family-Identifier (SUBAFI).

Configuring a routing protocol which supports multiple families consists of the following steps:

- Configuring global variables for the routing protocol
- Configuring variables specific to each address family
- Defining sessions with other routing peers
- Activating routing peers to carry a particular address family

The address families supported vary with the routing protocol. For example, under the current code BGP supports the following address families

- IPv4 unicast[1]
- IPv4 multicast[2]
- VPN-IPv4 unicast

2.5 VPN IPv4 NLRIs

As mentioned above, addresses within a VPN are not required to be unique. In order to carry these addresses across the provider backbone, we introduce a new address family known as 'VPN-IPv4' or 'VPNv4'. A VPNv4 prefix consists of

- an IPv4 prefix (32 bits, and a mask length)
- an 8 byte value known as a Route Distinguisher, or RD.

The VPNv4 prefix is created by pre-pending the RD to the VPN IPv4 prefix. Therefore, as long as a unique RD value is chosen the resultant VPNv4 prefix will be unique. To be precise, the same RD should be used only if two VPNv4 prefixes should be directly comparable.

An RD can be entered in two formats:

- A 16 bit ASN, followed by an arbitrary 32 bit number
- A 32 bit IP address, followed by an arbitrary 16 bit number

For example

- <ASN>:<number> or 100:1000
- <Address>:<number> or 128.90.1.2:1000

2.6 Controlling route distribution: VPN Route Target communities

The mechanism by which BGP/MPLS VPN controls distribution of VPN routing information is through the use of VPN **route-target** *extended BGP communities*. An extended BGP community is a 64 bit structured value, the format of which is defined in draft-ramachandra-bgp-ext-communities-00.txt.

BGP/MPLS VPN uses VPN route-target communities as follows:

---

1. Here 'IPv4' is the AFI, and 'unicast' is the SUBAFI.
2. Not in the first EFT

- When a VPN route is injected into BGP, it is associated with a list of VPN route-target communities. Typically this is set through an *export* list of community values associated with the VRF from which the route was learned.
- Associated with each VRF is an *import* list of route-target communities. This list defines the values which should be matched against to decide whether a route is eligible to be imported into this VPN routing instance. For example, if the import list for a particular VRF is **{A, B, C}**, then any VPN route which carries community value A, or B, or C will be imported into the VRF.

For examples of how these are used, consider the scenarios in the following sections.

2.6.1 Scenario 1: A closed user group VPN

The simplest scenario for a VPN is a closed group of sites. Every site can communicate directly with every other site. To support this scenario we need to do the following:

- define a single VPN route-target community, call it $\mathbf{C_{closed}}$
- Set the export list of each VRF connected to a site in the VPN to contain only $\mathbf{C_{closed}}$
- Set the import list of each VRF connected to a site in the VPN to contain only $\mathbf{C_{closed}}$

Thus, every route exported into IBGP from one of these VRFs will contain $\mathbf{C_{closed}}$. Every route which is received which contains $\mathbf{C_{closed}}$ will be eligible for import into these VRFs.

2.6.2 Scenario 2: a hub/spoke VPN

A more complex scenario would be a hub&spoke VPN. In this case we have one central site, the "hub", and a set of "spoke" sites. "spoke" sites can not communicate directly with each other; instead, all traffic between "spoke" sites will be routed through the "hub". The "hub" site can communicate directly to all "spokes". [1]

To implement this scenario we do the following:

- define two VPN route-target communities: $\mathbf{C_{hub}}$ and $\mathbf{C_{spoke}}$.
- Set the export list of each VRF connected to a "spoke" site to $\mathbf{C_{spoke}}$
- Set the export list of the "hub" site to $\mathbf{C_{hub}}$
- Set the import list of each VRF connected to a "spoke" site to $\mathbf{C_{hub}}$
- Set the import list of the "hub" site to $\mathbf{C_{spoke}}$

Thus, every route exported into IBGP from a "spoke" site is imported only into the VRF for the "hub" site. Conversely, routes exported from the "hub" site are imported into the VRFs for each "spoke" site.

This scenario can be generalized to multiple hub sites. Through the definition of additional route-target communities it can be generalized to support arbitrary overlap between the "spoke" sites.

2.6.3 Scenario 3: Controlled access to servers

As a third scenario, let us consider a case where we want control the access to a set of content servers. For example, assume we have 3 sites **A**, **B**, and **C**, and 3 servers $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$. Sites have subscribed to differing sets of content. In particular, **A** and **B** are allowed to receive data from $\mathbf{S_1}$; **B** and **C** from $\mathbf{S_2}$; and only **C** from $\mathbf{S_3}$. **A**, **B**, **C** may or may not be allowed to communicate directly.

To implement this scenario, we do the following:

---

1. Note: the current code does not support this configuration optimally. The current code requires that routes exported from a site be present in that site's VRF, otherwise MPLS forwarding information is not created correctly. Therefore, with the current code, additional route-target communities must be defined in order to cause the (e.g.) spokes routes to be imported into their own VRFs.
This example documents the "optimal" configuration, which will be supported in the future.

- define six route-target communities: C_{S1I}, C_{S1E}, C_{S2I}, C_{S2E}, C_{S3I}, C_{S3E}.
- Routes from server S_1 should be exported using community C_{S1E}. The VRF connected to server S_1 should be configured to import routes with community C_{S1I}. Similarly for the other two servers.
- Routes for site **A** should be exported with community C_{S1I}.
- Routes for site **B** should be exported with communities C_{S1I} and C_{S2I}.
- Routes for site **C** should be exported with communities C_{S2I} and C_{S3I}
- The VRF connected to site **A** should import routes with community C_{S1E}
- The VRF connected to site **B** should import routes with communities C_{S1E} and C_{S2E}
- The VRF connected to site **C** should import routes with communities C_{S2E} and C_{S3E}

3.0 Configuration Commands

3.1 Router Address Families

Router address families are configured by entering 'address-family' sub-mode, under router configuration sub-mode. This is done through the 'address-family' command. The general form of this command is:

```
(config)# router <router> <arguments>
(config-router)# ! global router variables
(config-router)#! neighbor session definition commands
(config-router)# address-family <afi> [ <subafi> ] [ vrf <name> ]
(config-router-af)# ! address-family specific router commands
(config-router-af)# exit-address-family ! return to router mode
```

The exact set of commands available under an address family will depend on the routing protocol being configured, and the address family.

3.1.1 BGP address families

BGP supports address families 'ipv4 unicast', 'ipv4 multicast' and 'vpnv4 unicast'

```
(config-router)# address-family ipv4 [ unicast ]
(config-router)# address-family ipv4 multicast
config-router# address-family vpnv4 [ unicast ]
```

3.1.2 BGP session commands

The following BGP commands are entered in router config mode, to define BGP sessions

```
(config-router)# neighbor {<address> | <peer-group>} remote-as <asn>
(config-router)# neighbor {<address> | <peer-group>} update-source <interface>
(config-router)# neighbor <peer-group> peer-group
(config-router)# neighbor <address> peer-group <peer-group>
```

3.1.3 Activating a BGP session for an address family

By default, a BGP session will carry IPv4 NLRIs for the global routing table. This behavior can be changed through the configuration knob

```
(config-router)# no bgp default ipv4-activate
```

If this command is given, then by default a BGP session will carry no address-families.

In order to activate a session for a particular address-family, the following command should be entered under the appropriate address-family sub-mode

```
(config-router-af)# [no ] neighbor { <address> | <peer-group> } activate
```

3.1.4 BGP global commands

The following configuration commands currently affect all address-families, and should be entered at the router config level.

```
(config-router)# bgp always-compare-med
(config-router)# bgp bestpath ...
(config-router)# bgp client-to-client reflection
(config-router)# bgp cluster-id ...
(config-router)# bgp confederation ...
(config-router)# bgp default local-preference ..
(config-router)# bgp deterministic-med
(config-router)# bgp fast-external-fallover
(config-router)# bgp log-neighbor-changes
(config-router)# bgp redistribute-internal
(config-router)# bgp router-id ...
(config-router)# timers bgp ...
```

3.1.5 BGP commands for address family IPv4

All BGP configuration commands which are supported in previous versions of IOS are valid for address-family IPv4 unicast. These affect either all IPv4 instances, or the default IPv4 routing table. Note for backward compatibility these commands can be entered either at router config mode, or in address-family mode for 'ipv4 unicast'.

3.1.6 BGP commands for address family VPNv4

The following configuration commands are supported under address-family vpnv4

```
(config-router-af)# bgp dampening ...
(config-router-af)# neighbor ....
(config-router-af)# neighbor { <address> | <peer-group> } activate
```

3.2 Defining a VRF

The first step in setting up VPN routing/forwarding is to define the instance. This is done through the configuration command

```
(config)# [no] ip vrf <name>
```

This command creates a VPN Routing table, and associated CEF table, named *<name>*. It then enters "vrf configuration" submode to configure variables associated with the VRF.

The prompt for VRF configuration submode is:

```
(config-vrf)#
```

3.2.1 Defining the default RD for a VRF

The VRF mode configuration command

(config-vrf)# rd <value>

Is used to set the default Route Distinguisher value for the VRF to be *<rd>*.

In the current VPN code[1], all routes from this VRF, which are injected into BGP will be converted to VPNv4 prefixes using this RD value.

It is necessary to specify an RD value before the VRF can be referenced in other IOS configuration and EXEC commands.

3.3 Associating interfaces with a VRF

Once a VRF has been defined, it can be associated with an interface through the interface level command

```
(config-if)# ip vrf forwarding <name>
```

This command is also valid on sub-interfaces.

Note that changing VRF forwarding for an interface will remove the IP address information configured for the interface. It is assumed that IP addressing information is not valid across different routing tables.

3.4 Configuring router variables for a VRF

To configure routing for a VRF, one uses the 'address-family' sub-mode, with a VRF qualifier. For example:

```
(config-router)# address-family ipv4 vrf <name>
(config-router-af)# ! Commands will be interpreted as applying to the named VRF
```

3.4.1 BGP router variables

The following variables can be configured for BGP under a VRF address-family sub-mode

```
(config-router-af)# aggregate-address
(config-router-af)# auto-summary
(config-router-af)# default-information originate
(config-router-af)# default-metric ...
(config-router-af)# distance ...
(config-router-af)# distribute-list ...
(config-router-af)# network ...
(config-router-af# neighbor ...
(config-router-af)# redistribute ...
(config-router-af)# synchronization
(config-router-af)# table-map ...
```

The default setting for 'auto-summary' in VRF address-family sub-mode is OFF.

The default setting for 'synchronization' in VRF address-family sub-mode is OFF.

Important note: A BGP session which will be used for exchange of IPv4 NLRIs with a VRF neighbor (i.e. a CE router) must not be activated for exchange of IPv4 NLRIs with any non-VRF neighbor (e.g. for Internet routes). Therefore, you must either set 'no bgp default ipv4-activate', or else do 'no neighbor ... activate' after configuring the BGP session commands for that neighbor under router config mode.

3.4.2 RIP router variables

The following variables can be configured for RIP under a VRF address-family sub-mode.

```
(config-router-af)# auto-summary
(config-router-af)# default-information originate
(config-router-af)# default-metric ...
(config-router-af)# distance ...
(config-router-af)# network ...
```

1. In the future, route-map extensions will be available to control the RD value

```
(config-router-af)# offset-list ...
(config-router-af)# redistribute ...
```

3.5 Configuring static routes for a VRF

In order to configure a static route for a VRF, use the 'ip route' configuration mode command with a 'vrf" keyword

```
(config)# ip route [ vrf <name> ] ...
```

3.6 Configuring IBGP for exchange of VPNv4 NLRIs

In order to configure an IBGP session between PE routers, or between a PE and a route-reflector, for the exchange of VPNv4 NLRIs one executes the following steps

- Define an IBGP BGP session in router config mode

```
(config-router)# neighbor <address> remote-as <asn>
(config-router)# neighbor <address> update-source <interface>
```

Important note: in order for VPN packets to be properly tag forwarded between the PE routers, it is important to use a loopback interface address for the neighbor address, and the update-source.

- Activate the session of VPNv4 NLRIs

```
(config-router)# address-family vpnv4
(config-router-af)# neighbor <address> activate
```

3.7 Configuring VPN route-target communities

3.7.1 Setting the export list for a VRF

The export VPN route-target list for a VRF defines the set of VPN route-target communities which will, by default[1], be attached to any routes from this VRF imported into VPN-IPV4 NLRIs in BGP. This is set in VRF configuration mode through the configuration command

```
(config-vrf)# [no]route-target export <value>
```

The <name> parameter specifies the VRF. The <value> takes two formats:

- A 16 bit ASN, followed by an arbitrary 32 bit number
- A 32 bit IP address, followed by an arbitrary 16 bit number

To add multiple VPN route-target communities to the list, issue the command multiple times.

To remove a VPN route-target community from the list, use the `no` form of the command.

3.7.2 Setting the import list for a VRF

The import VPN route target list for a VRF defines the set of VPN route-target which allow routes to be imported into a VRF. If any of the community values in the list are associated with a BGP VPN-IPv4 NLRI, then the IP address which corresponds to that NLRI is eligible[2] for redistribution into the VRF routing table. To set the import VPN route-target list for a VRF, use theVRF configuration mode command

```
(config-vrf)# [no]route-target import<value>
```

To add multiple VPN route-target communities to the list, issue the command multiple times.

---

1. Route map support for setting extended community attributes has not yet been implemented. When it is, setting route-target community values through a route map will override the default export route-target list for a VRF.
2. See Section 3.7.4

To remove a VPN route-target community from the list, use the **no** form of the command.

3.7.3 Setting both the import and export list for a VRF

In some cases, the same VPN route-target community value may need to be added to both the import and export list for a VRF. In this case, the following command can be used in VRF configuration mode:

```
(config-vrf)# [no]route-target both <value>
```

3.7.4 Setting the import route-map for a VRF

Once a route has been selected as eligible for import into a VRF, further control can be applied through the use of the, optional, VRF **import route-map**. This route map is applied after routes have been selected for import into a VRF. In other words, the algorithm used is the following:

For each VPN-IPv4 NLRI received, check the set of route-target communities against the import lists for all VRF's.

- For each VRF which matches
  - If there is an import route-map configured for the VRF, apply the route-map.
  - If the route-map denies access, then do not import the route
  - Otherwise, import the route

To specify the import route map, use the VRF configuration mode command

```
(config-vrf)# [no] import map <map name>
```

3.8 Other commands

3.8.1 Configuring whether CE-PE addresses are in the global routing table

In the current images, by default addresses on the link between the PE and CE routers are installed both in the appropriate VRF tables, and in the global routing/forwarding table. The following command can be used in VRF configuration mode to change this behavior:

```
(config-vrf)# [no]global-connected-addresses
```

The **no** form of this command changes the default behavior, and causes the addresses to be installed only in the VRF tables.

Note that a side effect of changing the state of 'global-connected-addresses' for a VRF is currently that all IP addressing information will be removed from all interfaces connected to the VRF.

4.0 EXEC Commands

4.1 Show VRF information

4.1.1 show VRFs and interface associations

To display the set of defined VRFs, and the interfaces associated with each, use the EXEC command:

```
router# show ip vrf [ <name> ]
```

4.1.2 Showing detailed VRF information

To display more information about a VRF, including the configured import and export community lists, use the EXEC command

```
router# show ip vrf [ <name> ] detail
```

4.1.3 show VRF routing information

To display the IP routing table for a VRF instance, use the EXEC command:

```
rcuter# show ip route [ vrf <name> ] ...
```

4.1.4 show VRF routing protocol information

To display the routing protocol information associated with a VRF, use the EXEC command

```
rcuter# show ip protocols [ vrf <name> ] ...
```

4.1.5 show VRF forwarding information

To display the CEF forwarding table associated with a VRF, use the EXEC command

```
rcuter# show ip cef [ vrf <name> ] ...
```

4.1.6 show VRF interface information

The following existing EXEC commands will display the VRF table associated with an interface

```
rcuter# show ip interface [ <interface> ]
rcuter# show cef interface [ <interface> ]
```

4.2 Show BGP VPNv4 information

To display VPNv4 information from the BGP database use the EXEC command

```
router# show ip bgp [ vpnv4 { all | rd <rd> | vrf <name> } ] ....
```

This will display VPNv4 NLRI information, qualified as follows:

- 'all' displays the complete database
- 'rd *<rd>*' displays NLRIs with a matching RD value
- 'vrf *<name>*' displays NLRIs associated with the named VRF.

To display the tags distributed with VPNv4 NLRIs, use the command:

```
router# show ip bgp [ vpnv4 { all | rd <rd> | vrf <name> } ] tags
```

This displays output of the form:

```
Network          Next Hop      In tag/Out tag
Route Distinguisher: 100:1 (vpn1)
   2.0.0.0          10.20.0.60      34/notag
   10.0.0.0         10.20.0.60      35/notag
   12.0.0.0         10.20.0.60      26/notag
                    10.20.0.60      26/notag
   13.0.0.0         10.15.0.15      notag/26
```

Where the 'In tag' information displays the tag (if any) assigned by this router, and the 'Out tag' information displays the tag assigned by the BGP Next Hop router.

4.3 show the tag forwarding tables for VRF routes

To display tag forwarding entries which correspond to VRF routes advertised by this router, use the EXEC command:

```
router# show tag-switching forwarding [ vrf <name> ] [ <prefix> <mask/length>] \
                                [ detail ]
```

4.4 Other commands

4.4.1 Clear VRF routing information

To clear routes from the VRF routing table, use the EXEC command:

```
router# clear ip route [ vrf <name> ] ...
```

4.4.2 Pinging in the context of a VRF

To ping from a PE router to addresses in the context of a particular VRF, use the command

```
router# ping [ vrf <name> ] ...
```

4.4.3 Traceroute in the context of a VRF

To traceroute from a PE router to addresses in the context of a particular VRF, use the command

```
router# traceroute [ vrf <name> ] ...
```

4.4.4 Telnet in the context of a VRF

To telnet from a PE router to addresses in the context of a particular VRF, use the command

```
router# telnet <address> [ <port> ] [ /vrf <name> ] ...
```

5.0 Configuration Examples

5.1 Basic configuration

The following is basic configuration file from a PE router, illustrating most of the configuration commands above.

```
! CEF switching is a pre-requisite for Tag
ip cef distributed
frame-relay switching
!
! Define two VPN Routing instances, named 'vpn1' and 'vpn2'
ip vrf vpn1 rd 100:1
ip vrf vpn2 rd 100:2
! Configure the import and export VPN route-target list for each VRF
ip vrf vpn1 route-target both 100:1
ip vrf vpn2 route-target both 100:2
ip vrf vpn2 route-target import 100:1
! Configure an import route-map for vpn2
ip vrf vpn2 import map vpn2_import
! 'vpn2' should not install PE-CE addresses in the global routing table
no ip vrf vpn2 global-connected-addresses
!
interface lo0
 no ip address
 ip address 10.13.0.13 255.255.255.255
 no shut
! Backbone link to another Provider router
interface atm9/0/0
  no shut
!
```

```
interface atm9/0/0.1 tag-switching
 no shut
 tag-switching ip
 ip unnumbered lo0
!
! Set up an Ethernet interface as a VRF link to a CE router
interface Ethernet5/0/1
 ip vrf forwarding vpn1
 ip address 10.20.0.13 255.255.255.0
 no shutdown
!
! Set up a Frame-Relay PVC sub-interface as another link to a CE router
interface hssi 10/1/0
  hssi internal-clock
  encaps fr
  frame-relay intf-type dce
  frame-relay lmi-type ansi
  no shut
!
interface hssi 10/1/0.16 point-to-point
  ip vrf forwarding vpn2
  ip address 10.20.1.13 255.255.255.0
  frame-relay interface-dlci 16
  no shut
!
! Configure BGP sessions
router bgp 1
! Define an IBGP session with another PE
  neighbor 10.15.0.15 remote-as 1
  neighbor 10.15.0.15 update-source lo0
  no synchronization
! Define some VRF (CE) sessions.
  neighbor 10.20.1.11 remote-as 65535
  neighbor 10.20.1.11 update-source h10/1/0.16
! Deactivate the default IPv4 session
  no neighbor 10.20.1.11 activate
  neighbor 10.20.0.60 remote-as 65535
  neighbor 10.20.0.60 update-source e5/0/1
  no neighbor 10.20.0.60 activate
!
! Activate PE peer for exchange of VPNv4 NLRIs
  address-family vpnv4 unicast
   neighbor 10.15.0.15 activate
```

```
  exit-address-family
!
! Define router variables for VRFs
! Activate sessions to peers within those VRFs
  address-family ipv4 unicast vrf vpn1
   neighbor 10.20.0.60 activate
   no auto-summary
   redistribute static
  exit-address-family

  address-family ipv4 unicast vrf vpn2
   neighbor 10.20.1.11 activate
   no auto-summary
   redistribute static
  exit-address-family
!
! Define a VRF static route
ip route vrf vpn1 12.0.0.0 255.0.0.0 e5/0/1 10.20.0.60
```

5.2 Configuring hub/spoke VPN

As a further example, the following illustrates how one might configure the hub/spoke VPN discussed in Section 2.6.2.

```
ip vrf hub rd 1.2.3.4:1
ip vrf spoke rd 1.2.3.4:2
ip vrf hub route-target export 1.2.3.4:2
ip vrf hub route-target import 1.2.3.4:1
ip vrf spoke route-target export 1.2.3.4:1
ip vrf spoke route-target import 1.2.3.4:2
```

5.3 Configuring directly connected hosts

Configuring a PE to support directly connected hosts, for example servers, is essentially the same. The configuration required is:

- define the VRFs
- connect the interfaces to the correct VRFs
- ensure that the connected address is distributed into the VRF BGP instance, either through a **`network`** statement, or through **`redistribute connected`**.

5.4 Configuring VPN VPDN termination

In some cases, one might want to use a PE router to terminate a VPDN tunnel link. For example, the scenario in [Figure 3,].

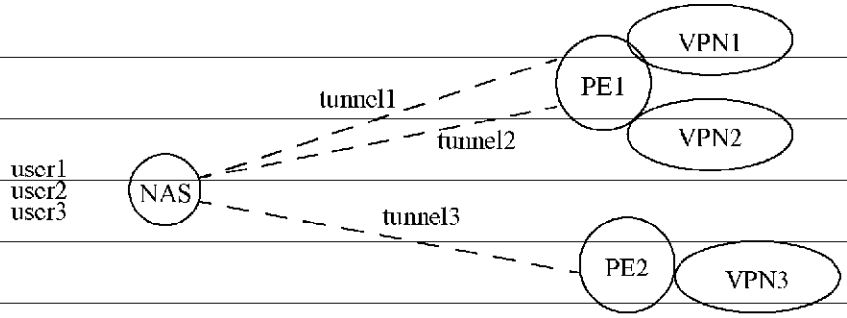

Figure 3, VPN dial access

In this scenario we have a Network Access Server (NAS) which serves customers for a number of different VPNs. Based on the authentication information supplied by each user, the NAS directs the user's traffic to a VPDN tunnel[1].

Each tunnel is terminated on a PE "Virtual Home Gateway" router. Associated with each tunnel is:

- A VRF
- An address pool for assignment of addresses to the dial-in users

On the PE, BGP is configured to distribute aggregate VPN-IPv4 addresses which cover the address pool for each tunnel.

This scenario can be accomplished by using the following configuration commands on the NAS and PE

NAS:

```
! Map users to the correct tunnels
vpdn enable
vpdn outgoing cisco nas-cisco ip <PE1 address>
vpdn outgoing descend nas-descend ip <PE1 address>
```

PE:

```
!Configure VPDN termination
vpdn enable
vpdn incoming nas cisco pe1 virtual template 1
vpdn incoming nas-descend pe1 virtual-template 2
!
!Create two VRFs for two dial in VPN communities
ip vrf CISCO rd 100:1
```

1. In the current VPN code base, only L2F is supported as a VPDN tunneling protocol

```
ip vrf DESCEND rd 100:2
ip vrf CISCO route-target both 100:1
ip vrf DESCEND route-target both 100:2
! Define two Virtual-Template interfaces
interface virtual-template 1
 ip vrf forwarding CISCO
 ip address 10.100.0.1 255.255.255.0
 no ip directed-broadcast
 ppp authentication ms-chap
peer default ip address pool CISCO
!
interface virtual-template 2
 ip vrf forwarding DESCEND
 ip address 10.200.0.1 255.255.255.0
 no ip directed-broadcast
 ppp authentication ms-chap
peer default ip address pool DESCEND
!
! Define address pools for each dial in group
ip local pool CISCO 10.100.0.2 10.100.0.254
ip local pool DESCEND 10.200.0.2 10.200.0.254
!
! Configure routing for the two VRFs
router bgp 1
  address-family ipv4 vrf CISCO
    network 10.100.0.0 mask 255.255.255.0
    aggregate 10.100.0.0 255.255.255.0 summary
    redistribute connected
  exit-address-family
address-family ipv4 vrf DESCEND
    network 10.200.0.0 mask 255.255.255.0
    aggregate 10.200.0.0 255.255.255.0 summary
    redistribute connected
  exit-address-family
```